Guide de formation avec cas pratiques

Excel 2013 Programmation VBA

Daniel-Jean David

Tsoft EDITEUR

EYROLLES

© Tsoft et Groupe Eyrolles, 2014, ISBN : 978-2-212-13905-1

Vie d'un programme

2

Différentes façons de lancer une procédure

Mise au point d'une macro

Utiliser l'aide

L'explorateur d'objets

Récupération des erreurs

DIFFÉRENTES FAÇONS DE LANCER UNE PROCÉDURE

PAR INSTRUCTION D'APPEL

Toute procédure peut être appelée depuis une autre procédure (ou fonction) par l'instruction d'appel de la forme :

```
[Call] <nom de la proc. appelée> [<arguments éventuels>]
```

Exemples :

```
Traitement          'il n'y a pas d'arguments
Calcul 5, 4         '2 arg. ; procédure supposée définie par :
                    'Sub Calcul (a as Integer, b as Integer)
```

Le mot-clé `Call` n'est presque jamais présent. Notez que la liste des arguments est entre parenthèses () dans la déclaration de la procédure, et sans parenthèse () dans l'appel. Les parenthèses () dans l'appel caractérisent une fonction ; si vous les mettez alors qu'il y a plusieurs arguments, il faut utiliser `Call`. Pour plus de détails sur ces points, voyez le chapitre *Procédures, fonctions, arguments*.

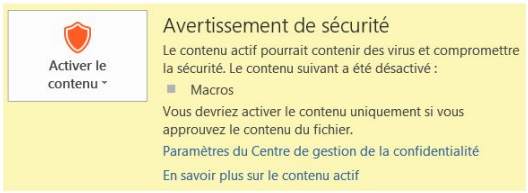
Cette manière de lancer une procédure est dite « méthode interne », mais elle pose question : comment lancer la procédure appelante. On voit qu'il faut des méthodes « externes ».

PAR MENUS STANDARDS

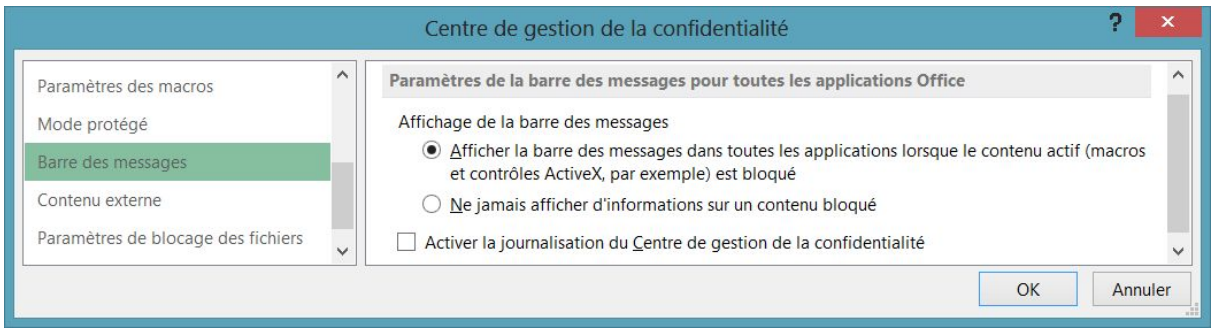
Depuis le classeur Excel

Lorsqu'on ouvre un classeur Excel qui contient des macros, le comportement varie en fonction du niveau de sécurité choisi dans les options. Avec le niveau le plus conseillé, vous avez une barre en haut de l'écran :

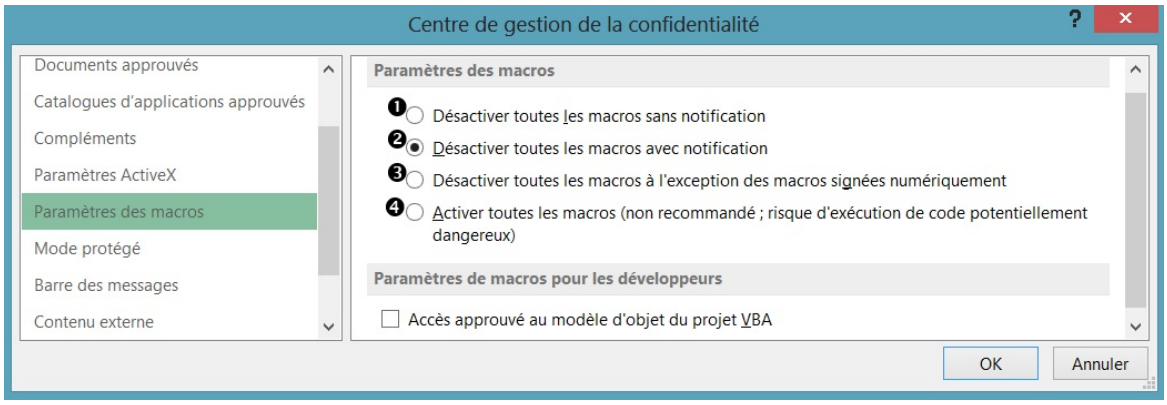
- Si vous actionnez Activer le contenu vous pourrez essayer les macros que vous aurez créées.
- Si vous actionnez *Les macros ont été désactivées*, la BDi de 🗀*FICHIER -Informations* s'affiche avec une partie d'avertissement :


Activer le contenu
Avertissement de sécurité
Le contenu actif pourrait contenir des virus et compromettre la sécurité. Le contenu suivant a été désactivé :
Macros
Vous devriez activer le contenu uniquement si vous approuvez le contenu du fichier.
Paramètres du Centre de gestion de la confidentialité
En savoir plus sur le contenu actif

- Actionnez *Paramètres du Centre de gestion de la confidentialité*. vous obtenez la même BDi que celle de 🗀*FICHIER – Options – Centre de gestion de la confidentialité* – Paramètres du Centre de gestion de la confidentialité. Son onglet *Barre des messages* doit être ainsi :


Centre de gestion de la confidentialité
Paramètres des macros
Mode protégé
Barre des messages
Contenu externe
Paramètres de blocage des fichiers
Paramètres de la barre des messages pour toutes les applications Office
Affichage de la barre des messages
Afficher la barre des messages dans toutes les applications lorsque le contenu actif (macros et contrôles ActiveX, par exemple) est bloqué
Ne jamais afficher d'informations sur un contenu bloqué
Activer la journalisation du Centre de gestion de la confidentialité
OK
Annuler

- Son onglet *Paramètres des macros* fixe le niveau de sécurité :

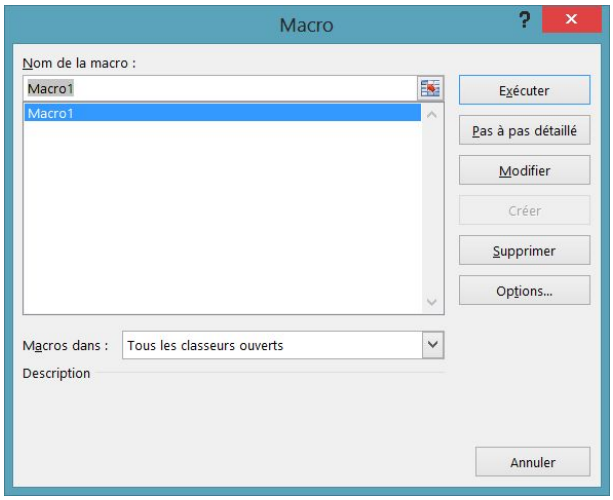

Centre de gestion de la confidentialité
Documents approuvés
Catalogues d'applications approuvés
Compléments
Paramètres ActiveX
Paramètres des macros
Mode protégé
Barre des messages
Contenu externe
Paramètres des macros
❶ Désactiver toutes les macros sans notification
❷ Désactiver toutes les macros avec notification
❸ Désactiver toutes les macros à l'exception des macros signées numériquement
❹ Activer toutes les macros (non recommandé ; risque d'exécution de code potentiellement dangereux)
Paramètres de macros pour les développeurs
Accès approuvé au modèle d'objet du projet VBA
OK
Annuler

Le comportement que nous préconisons est obtenu avec l'option ❷. L'option ❶ est pour les utilisateurs trop prudents. Si vous n'utilisez que les macros de ce livre, vous pouvez choisir l'option ❸, ce qui vous évitera d'activer les macros à chaque ouverture de classeur. Ce livre ne vous apprendra pas à créer des macros à virus. Donc, le risque évoqué ici ne devrait pas trop nous effrayer. En revanche, avant tout essai de vos « œuvres », il est impératif que vous sauvegardiez le classeur, car il y a un risque réel de blocage de l'ordinateur suite à une erreur dans une macro VBA, même les exemples de ce livre : vous n'êtes pas à l'abri des fautes de frappe.

- Si vous utilisez aussi des classeurs « étrangers », choisissez l'option ❷, car ❹ n'affiche aucune notification.

Ensuite, on peut choisir la procédure à exécuter :

- Faites 🗀*DÉVELOPPEUR – [Code] – Macros* ; la BDi de choix de macro montre la liste de toutes les procédures dans les classeurs ouverts.
- Choisissez dans la liste déroulante *Macros dans* le domaine où chercher les macros, soit l'un des classeurs ouverts, soit tous.
- Cliquez sur la macro/procédure voulue et Exécuter.



Depuis l'éditeur VBA

- Étant dans l'écran de VBA, faites afficher la fenêtre de module voulue si elle ne l'est pas déjà.
- Dans cette fenêtre, placez le curseur texte n'importe où à l'intérieur de la procédure voulue (entre `Sub` et `End Sub`).
- *Exécution – Exécuter Sub/User Form* ou touche de raccourci F5.

Attention : cette méthode fait exécuter la procédure de la même façon que la précédente sauf qu'il peut n'y avoir aucune feuille ni cellule active, alors que depuis la fenêtre Excel, ces éléments étaient définis. Donc si l'écriture de la procédure fait des hypothèses sur ces données, le fonctionnement risque d'être incorrect. Le mieux serait de rendre la rédaction indépendante en ajoutant des instructions pour activer la feuille et la cellule voulues.

PAR ÉVÉNEMENTS

Tout événement (clic, déplacement ou autre) peut être associé à une procédure qui sera exécutée à la survenance de l'événement. Si on fournit une procédure elle sera exécutée à l'arrivée de l'événement avant (ou, si la procédure le spécifie, à la place de) l'action standard du système pour cet événement. Cette action système peut être rien, auquel cas, si vous ne fournissez pas de procédure, votre application sera insensible à cet événement.

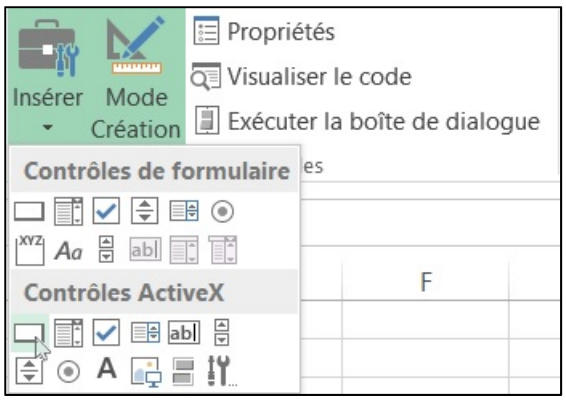
On distingue les *événements naturels* qui arrivent dans tout classeur (ex. changement de valeur dans une cellule, déplacement de la cellule active, activation d'une feuille, ouverture d'un classeur, passage d'un contrôle à un autre dans une BDi, validation d'une BDi…) et les *événements ad-hoc* qui sont introduits uniquement pour démarrer une certaine procédure par un simple clic, ce qui est beaucoup moins fastidieux que la méthode précédente.

Événements ad-hoc

On va créer un élément : bouton, forme géométrique, image, bouton de barre d'outils, nouveau menu ou nouvelle rubrique de menu et à l'événement clic sur cet élément on va associer la procédure que nous voulons lancer facilement. La personnalisation des barres d'outils et menus est discutée dans le chapitre *Commandes par boutons, barres d'outils ou menus*. Ici, nous ne regardons que le cas des boutons ou des dessins.

Pour implanter un contrôle bouton :

- Affichez la boîte à outils *Contrôles* par 🗀*DÉVELOPPEUR – [Contrôles] – Insérer* :



- Choisissez l'outil *Bouton de commande (Contrôle ActiveX)*.
- Le curseur souris prend la forme d'une croix ; délimitez le rectangle du bouton par glissement souris sur la feuille.
- Cliquez droit sur le bouton ; choisissez *Propriétés* dans le menu déroulant. Il apparaît une fenêtre de propriétés analogue à celle de VBA. Le plus indispensable est de changer la propriété *Caption* (libellé qui s'affiche sur le bouton) pour remplacer le libellé passe-partout *CommandButton1* par une mention spécifique du traitement (Ex. *Nouveau Client*…)

- Fermez la fenêtre de propriétés. Nouveau clic droit sur le bouton et *Visualiser le code*. On passe alors à la fenêtre VBA et dans un module intitulé du nom de la feuille de calcul où se trouve le bouton on trouve l'enveloppe d'une procédure *CommandButton1 _Click*. Il suffit d'y taper l'appel de la procédure à associer, c'est-à-dire son nom.
- Quittez le mode création par clic sur 🗀*DÉVELOPPEUR – [Contrôles] – Mode Création*.

Pour implanter un contrôle dessin :

- Faites 🗀*INSERTION – [Illustrations] – Formes*. Cliquez sur un rectangle (vous pouvez aussi choisir l'ellipse ou *Image*).
  - Pour rectangle ou ellipse, délimitez le rectangle conteneur par glissement souris sur la diagonale. Clic droit sur le contrôle, *Ajouter du texte* dans le menu déroulant, tapez le texte voulu. Puis clic droit vous permet d'agir sur la police (nous suggérons de choisir *Gras*) et *Format de la forme* ; regardez plus particulièrement *Zone de texte* qui permet de spécifier l'alignement (nous suggérons *Centré* pour Horizontal et Vertical) et *Remplissage* où nous suggérons de spécifier un remplissage gris clair. Vous pouvez agir aussi sur l'épaisseur de bordure. Le groupe 🗀*FORMAT – [Styles de forme]* offre beaucoup de possibilités.
  - Pour une image faites 🗀*INSERTION – [Illustrations]* et choisissez le fichier voulu dans la BDi qui apparaît. L'image vient en superposition sur la feuille : ajustez sa taille et faites la glisser à l'emplacement souhaité.
- Clic droit sur le contrôle et *Affecter une macro* dans le menu déroulant. Une BDi de choix de macro quasi identique à la figure du début de chapitre apparaît.
- Choisissez la procédure voulue et OK.
  Si au lieu de choisir une procédure existante dans la liste vous gardez le nom proposé d'emblée (exemple : *Rectangle1_QuandClic*), on passe dans l'éditeur VBA ce qui vous permet de taper le contenu de la procédure dans le Module 1.

Événements naturels

Ce sont les événements pour lesquels il n'y a pas besoin de créer un objet à cliquer. Ces événements peuvent se produire d'office. Si vous ne fournissez pas de procédure affectée à un tel événement, c'est l'action normale du système qui prévaut. Si vous fournissez une procédure, elle est exécutée avant l'action système et elle peut éventuellement l'inhiber.

Ces procédures doivent être placées dans la fenêtre de code du module associé au conteneur de l'objet concerné :

- pour un contrôle d'une BDi, c'est le module de code de la BDi.
- pour une cellule ou une zone de feuille de calcul, c'est le module associé à la feuille : vous ouvrez un tel module par double-clic sur *Feuil<n>* dans l'arborescence *Microsoft Excel Objects* ; pour un élément concernant le classeur entier, c'est *ThisWorkbook* dans la même arborescence. Ces fenêtres de code ont en haut deux listes déroulantes.

Pour définir une telle routine, choisissez l'objet dans la liste de gauche, puis la routine dans la liste de droite. Principaux objets et événements :

Conteneur	*Objet*	*Événements*
Feuille	Contrôle	`CommandButton<n>_Click` : clic sur le contrôle (ex. bouton)
"	Worksheet	`WorkSheet_SelectionChange` : on active une autre cellule
"	"	`Worksheet_Change` : on change le contenu de cellule

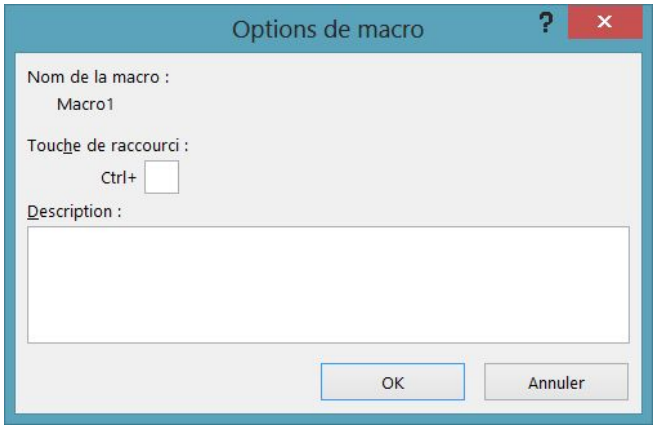
Conteneur	*Objet*	*Événements*
Classeur entier	Workbook	`Workbook_Activate` : activation du classeur
"	"	`Workbook_Open` : ouverture du classeur (*)
"	"	`Workbook_BeforeClose` : avant fermeture du classeur
BDi	Contrôle	`<Contrôle>_Click` : clic sur le contrôle
"	"	`<Contrôle>_Enter` : on arrive sur le contrôle
"	"	`<Contrôle>_Exit` : on quitte le contrôle
"	"	`<Contrôle>_Change` : on change la valeur du contrôle

`(*) Workbook_Open` permet d'implanter un traitement qui se fera dès qu'on ouvrira le classeur ; c'est le moyen d'assurer le **démarrage automatique** d'une application.

PAR RACCOURCI CLAVIER

Une autre solution semble très séduisante : on peut associer une combinaison Ctrl + Touche au déclenchement de l'exécution. On peut spécifier la touche dans la BDi d'enregistrement de la macro. Il faut y penser juste avant l'enregistrement. Si vous n'y avez pas pensé ou s'il s'agit d'une procédure entrée directement par l'Éditeur :

- *DÉVELOPPEUR – [Code] – Macros* : La BDi (de la page 27) apparaît. Attention, il faut demander cette commande depuis la fenêtre Excel, et non VBA ; depuis VBA, la commande *Outils – Macros* fait apparaître la même BDi, mais sans bouton Options.
- Choisissez la procédure voulue.
- Options : La BDi suivante apparaît (elle permet aussi de fournir une description).



L'inconvénient à notre avis rédhibitoire de ce dispositif est que si vous choisissez une combinaison qui a déjà une fonction, celle-ci disparaît et le système ne prévient absolument pas. Vous risquez ainsi de perdre irrémédiablement un raccourci extrêmement important.

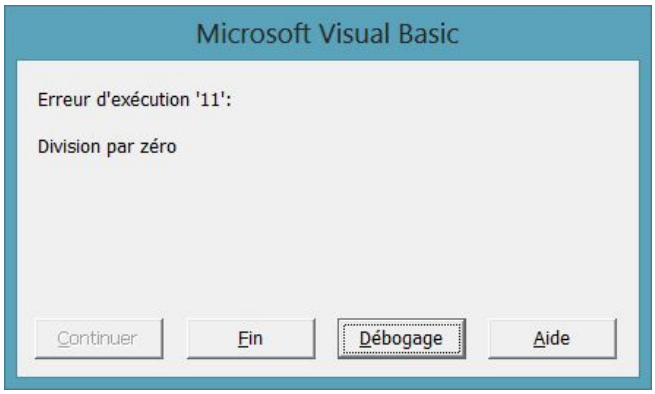
Une alternative plus intéressante est offerte par l'événement `OnKey` de l'objet `Application`. Il offre même plus de possibilités : on n'est pas limité aux combinaisons avec Ctrl et on peut rétablir l'ancienne fonction de la combinaison. Ceci est traité au chapitre 8.

Une fois écrite, la macro ne donne pas forcément du premier coup les résultats souhaités. Différents comportements sont possibles au moment où on demande l'exécution pour un premier essai (redonnons d'ailleurs ce conseil qu'on ne répétera jamais assez : sauvegardez le classeur avant de demander l'exécution) :

- le programme peut s'arrêter avant même d'avoir démarré en signalant une erreur de compilation (1) ;
- le programme s'arrête sur message d'erreur (2) ;
- le programme tourne indéfiniment (3) ;
- le programme s'achève, mais les résultats sont faux ; signalons que pour pouvoir déceler une telle erreur, il faut effectuer certains essais avec des données telles qu'on connaisse d'avance les résultats, ou qu'ils soient facilement calculables (4).

(1) Montre l'instruction en cause surlignée en jaune. Les erreurs de syntaxe concernées sont plus subtiles que celles qui sont décelées à l'écriture ; elles mettent souvent en jeu des incompatibilités entre plusieurs instructions alors qu'à l'écriture, l'analyse se limite à une instruction.
On peut faire apparaître ces erreurs en demandant *Débogage – Compiler VBAProject*.
L'avantage par rapport à l'exécution est que ceci détecte toutes les erreurs de syntaxe alors que l'exécution ne donne que celles des instructions par où on est passé.

(2) Fait apparaître une BDi comme :



et le programme se trouve arrêté. Si le bouton Débogage est présent (il est absent si l'écran VBA n'est pas activé), et si vous cliquez dessus, vous passez à l'affichage du module et l'instruction en cause est surlignée en jaune. Nous verrons plus loin ce qu'on peut faire.

(3) Est vraisemblablement dû à une portion de programme qui boucle. Le plus souvent, on arrive à reprendre le contrôle par la combinaison Ctrl + Pause. On est alors ramené au cas précédent : une des instructions de la boucle en cause est surlignée. On peut donc voir quelle est la boucle infinie et, de là, comprendre si la condition d'arrêt est mal exprimée ou si les données qui y interviennent sont mal calculées.

(4) Est le plus difficile à gérer puisque là, c'est la logique du programme qui est en cause. Les outils à mettre en œuvre sont les mêmes que pour les autres cas.

OUTILS DE MISE AU POINT

Les outils offerts par VBA pour aider à comprendre les erreurs sont, d'une part des moyens d'affichage (infobulles, fenêtre Variables locales, Pile des appels, Espions), d'autre part des moyens d'exécution (Pas à pas, Points d'arrêt, instruction Stop).

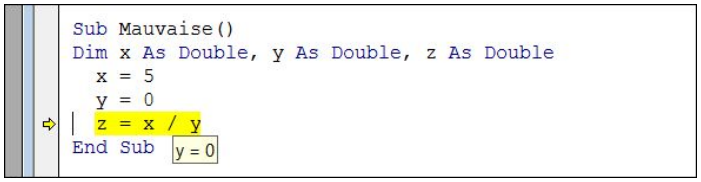
La fenêtre Exécution appartient aux deux catégories puisqu'on peut y afficher des données, mais aussi y taper des instructions. Ces moyens servent plus souvent en mode arrêt, mais certains peuvent être exploités pendant que le programme tourne et ce n'en est que mieux.

MOYENS D'AFFICHAGE

Infobulles

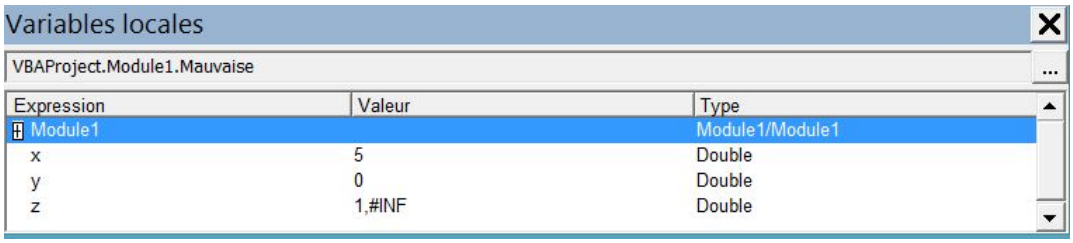
Lorsque le programme est arrêté sur erreur, si vous amenez le curseur souris sur une variable dans la procédure où on se trouve, il apparaît une info bulle qui donne la valeur.

L'exemple de code suivant qui sert à afficher le dialogue que vous voyez au bas de cette page ; si vous amenez le curseur souris sur `y`, vous obtenez une infobulle `y=0` :



Fenêtre variables locales

On l'obtient par *Affichage – Fenêtre Variables locales* dans l'écran VBA. Elle donne la valeur des variables :



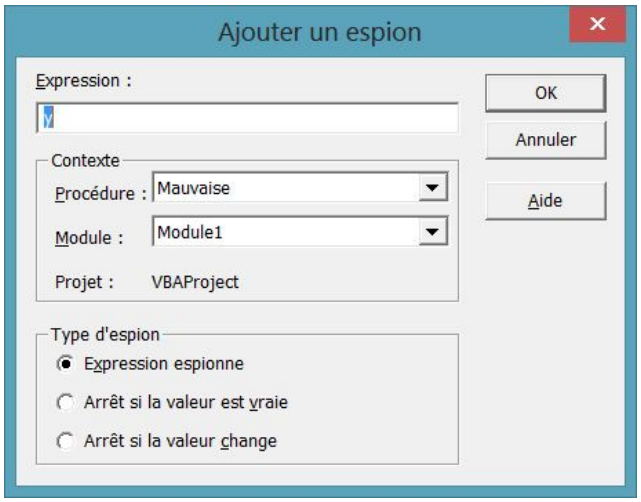
Un point très important est que vous pouvez modifier une valeur dans cette fenêtre : sélectionnez la valeur, modifiez-la puis cliquez ailleurs dans la fenêtre.

Pile des appels

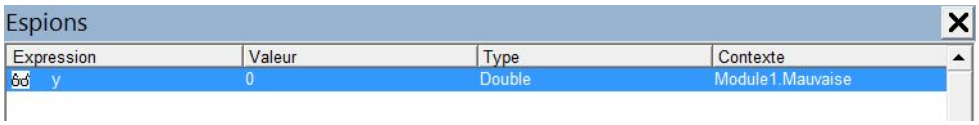
Un clic sur le bouton `...` à l'extrême droite de la ligne VBAProject…, ou *Affichage – Pile des appels* donne une fenêtre qui affiche la succession des appels de procédures. C'est utile dans les cas les plus complexes.

Espions

- Sélectionnez la variable *y*.
- *Débogage – Ajouter un espion*. *y* apparaît comme expression espionne :



Les choix les plus intéressants sont les boutons radio. Ils parlent d'eux-mêmes.



Espion express

Si vous avez oublié de définir un espion avant que le programme ne s'arrête sur erreur, il est encore temps :

- Sélectionnez l'expression voulue.
- Débogage – Espion express.

Un clic sur Ajouter ajoute l'expression comme espion.

MOYENS D'EXÉCUTION

Pas à pas

On peut demander l'exécution pas à pas, c'est-à-dire instruction par instruction. On l'obtient par *Outils – Macro – Macros*, puis Pas à pas détaillé depuis l'écran VBA (🗀 *DÉVELOPPEUR – [Code] – Macros* depuis Excel). Sinon, ayant le curseur souris dans la procédure voulue, demandez *Débogage – Pas à pas détaillé*.

Ceci est extrêmement fastidieux et ne doit être utilisé qu'en dernier ressort si on ne comprend pas la cause de l'erreur. Un peu moins fastidieux sont (*Débogage –*)*Pas à pas principal* (qui exécute les procédures appelées à vitesse normale) et *Pas à pas sortant* (qui fait sortir de la procédure en cours à vitesse normale). En mode pas à pas, on avance d'une instruction par F8.

Points d'arrêt

Il peut être préférable d'introduire quelques points d'arrêt, par exemple avant un passage qu'on voudra surveiller particulièrement. Pour cela :

- Amenez le curseur sur l'instruction voulue.
- *Débogage – Basculer le point d'arrêt* (Raccourci F9). Cette même commande permet d'ailleurs de supprimer le point d'arrêt. Un point d'arrêt apparaît sous forme d'un point bordeaux dans la marge grise.

Supprimer les points d'arrêt

Nous venons de voir comment en supprimer un. Pour supprimer tous les points d'arrêt, c'est *Débogage – Effacer tous les points d'arrêt* (Ctrl+Maj+F9).

Exécuter jusqu'au curseur

Une autre commande qui fait le même effet qu'un point d'arrêt (mais il ne peut y en avoir qu'un) est *Débogage – Exécuter jusqu'au curseur* Ctrl+F8. Il faut bien sûr avoir préalablement placé le curseur dans la fenêtre module sur l'instruction voulue.

Instruction Stop

Les points d'arrêt ne sont pas conservés lorsqu'on sauve le programme. On peut à la place insérer des instructions `Stop` qui font arrêter le programme de la même façon et permettent tout autant d'examiner les variables et les espions.

Que faire après un arrêt ?

Après avoir éventuellement modifié certaines données, on peut :

- continuer pas à pas à coups de F8.
- reprendre l'exécution là où on est ; cela se fait par *Exécution – Continuer* ou F5 ou ▶
- reprendre l'exécution à une autre instruction. Pour cela, il suffit de faire glisser à la souris la flèche jaune qui marque l'instruction où on en est dans la marge grise. Une autre manière est de cliquer sur l'instruction voulue puis *Débogage – Définir l'instruction suivante* ou Ctrl+F9.
- tout remettre à zéro, soit parce qu'on voudra ré-exécuter depuis le début, soit parce qu'on veut abandonner temporairement pour étudier le problème. Cela s'obtient par clic sur ■ ou *Exécution – Réinitialiser* ou clic sur Fin dans la BDi de la figure page 31. Cela peut aussi avoir lieu si vous modifiez le programme : une BDi vous prévient.

La fenêtre Exécution

En fait, la technique moins fastidieuse pour comprendre ce qui se passe dans un programme est de l'exécuter à vitesse normale, mais en insérant par endroits des ordres d'impression de données stratégiques. Pour cela, on peut utiliser `MsgBox`, mais cette instruction crée un arrêt ; exactement ce que nous voulons éviter. La solution est d'utiliser la fenêtre Exécution. Au lieu de `MsgBox <donnée>`, on utilise `Debug.Print <donnée>` et l'écriture se fera dans la fenêtre Exécution, sans causer d'arrêt. Les données à imprimer ainsi peuvent être des valeurs de variables, des textes du genre « On décèle l'événement … », ou « On arrive à la procédure … ».

Pour visualiser la fenêtre Exécution dans l'écran VBA, faire *Affichage – Fenêtre Exécution.* (Ctrl+G).

Le mode immédiat

Une particularité très intéressante de la fenêtre Exécution est que vous pouvez y taper des instructions VBA. Chaque instruction sera exécutée dès que vous taperez Entrée. C'est ce qu'on appelle le mode immédiat.

L'instruction la plus utilisée dans ce contexte est `Print` (abrégé : `?`) `<variable>`. Elle est intéressante car si l'on est en mode arrêt, les valeurs des variables avant l'arrêt sont connues, donc un `?<cette variable>` a autant d'efficacité que les espions et fenêtre Variables locales.

Par exemple, `?ActiveWorkbook.Name` donne le nom du classeur actif ; si ce n'est pas celui que vous avez prévu, vous avez bientôt compris pourquoi le programme ne fonctionne pas.

Vous pouvez aussi taper des instructions qui modifient des valeurs de variables ou des données dans les classeurs, et reprendre l'exécution avec les données modifiées à l'instruction que vous voulez.

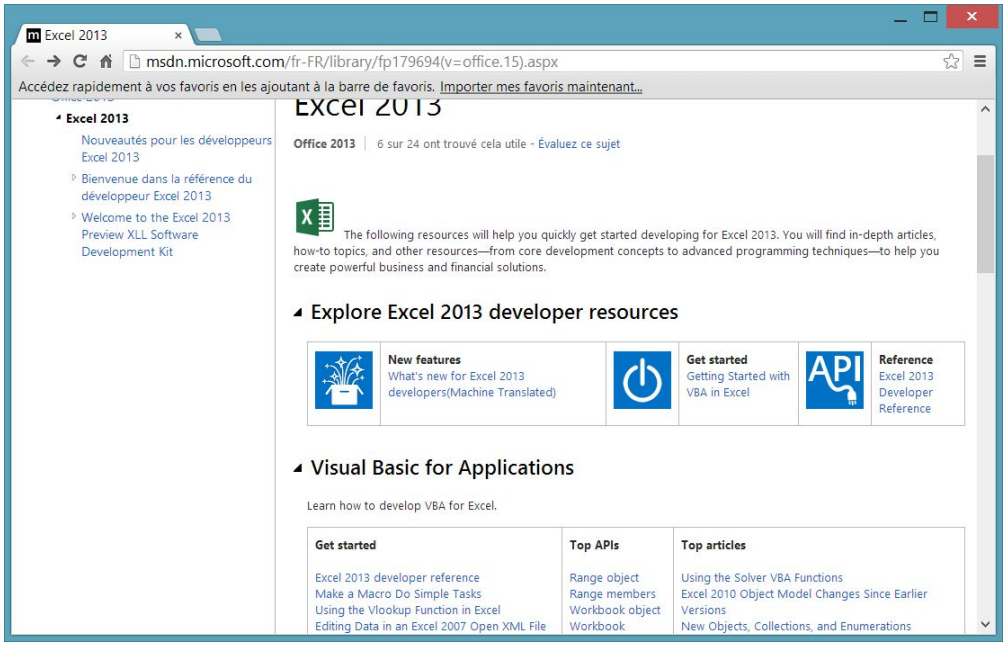
Une autre possibilité de la fenêtre Exécution est qu'elle permet d'essayer des instructions : vous tapez l'instruction à essayer dans la fenêtre Exécution et vous vérifiez les effets.

L'aide en ligne est un élément essentiel. Vous devez l'installer complètement. Si vous appartenez à une organisation où l'installation dépend du « Service Informatique », vous devez obtenir qu'il installe l'aide en ligne.

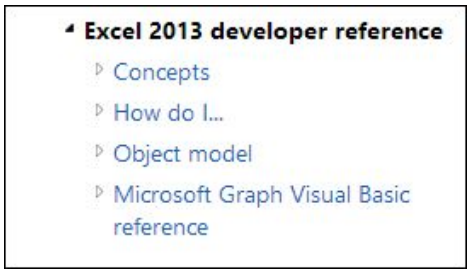
L'aide intervient déjà dans le fait de proposer automatiquement de compléter les instructions lors de leur écriture. De plus, si vous tapez F1 après un mot-clé ou alors qu'il est sélectionné, l'aide sur ce mot-clé apparaît. En outre, les BDi qui apparaissent lors d'un arrêt ont un bouton Aide qui amène à une page en rapport avec le problème.

Appel direct de l'aide

- Vous devez être dans l'écran VBA, sinon, c'est l'aide sur Excel que vous obtiendrez.
- ? – Aide sur Microsoft Visual Basic ou clic sur


Excel 2013
msdn.microsoft.com/fr-FR/library/fp179694(v=office.15).aspx
Accédez rapidement à vos favoris en les ajoutant à la barre de favoris. Importer mes favoris maintenant...
Excel 2013
Nouveautés pour les développeurs Excel 2013
Bienvenue dans la référence du développeur Excel 2013
Welcome to the Excel 2013 Preview XLL Software Development Kit
Excel 2013
Office 2013
6 sur 24 ont trouvé cela utile - Évaluez ce sujet
The following resources will help you quickly get started developing for Excel 2013. You will find in-depth articles, how-to topics, and other resources—from core development concepts to advanced programming techniques—to help you create powerful business and financial solutions.
Explore Excel 2013 developer resources
New features
What's new for Excel 2013 developers(Machine Translated)
Get started
Getting Started with VBA in Excel
Reference
Excel 2013 Developer Reference
Visual Basic for Applications
Learn how to develop VBA for Excel.
Get started
Excel 2013 developer reference
Make a Macro Do Simple Tasks
Using the Vlookup Function in Excel
Editing Data in an Excel 2007 Open XML File
Top APIs
Range object
Range members
Workbook object
Workbook
Top articles
Using the Solver VBA Functions
Excel 2010 Object Model Changes Since Earlier Versions
New Objects, Collections, and Enumerations

- Appelez ensuite *Excel 2013 Developer reference* qui propose les choix :



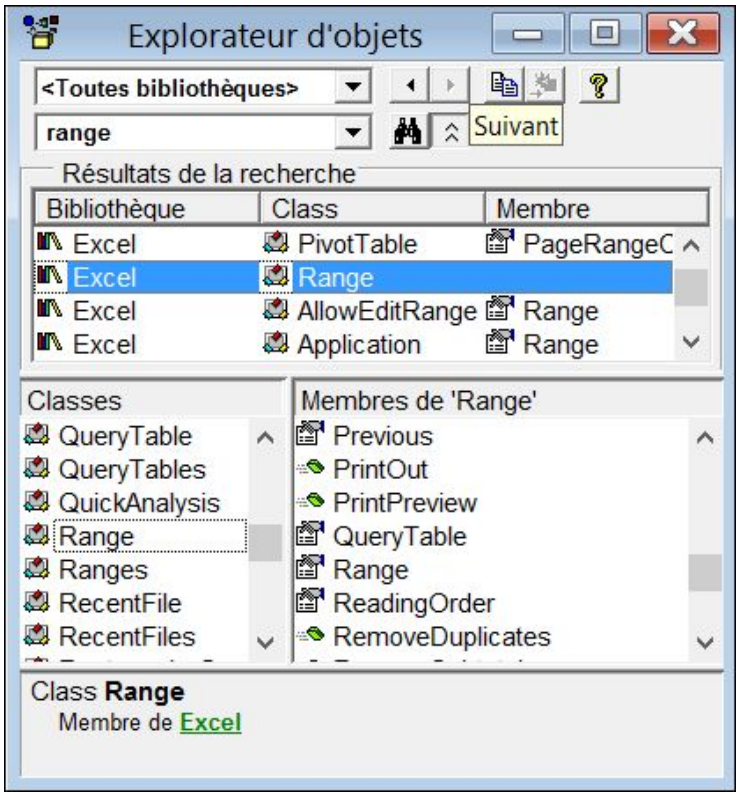
Nous n'insistons pas sur le mode d'emploi de la navigation qui est classique : on développe une arborescence en cliquant sur le livre fermé et on la résorbe en cliquant sur le livre ouvert. Sinon, c'est un hypertexte classique.

La zone d'entrée *Rechercher* sert à taper un mot et le système propose des rubriques ou demande de reformuler la question.

L'EXPLORATEUR D'OBJETS

L'Explorateur d'objets est une extraordinaire source de renseignements, d'autant que la programmation VBA est surtout dépendante des objets de l'application hôte (Excel dans notre cas).

- Dans l'écran VBA, faites *Affichage – Explorateur d'objets* (F2)



- Dans la première liste déroulante, choisir :
  - Soit <Toutes bibliothèques>
    - Tapez le mot cherché dans la 2e liste déroulante.
    - Choisissez ensuite une classe ou un membre sous *Résultats de la recherche*.
    - Vous pouvez alors choisir un membre dans la dernière liste. Le type d'un membre se reconnaît à l'icône devant son nom :

  - Soit une des bibliothèques, par exemple VBA.
    - Choisissez une classe dans la liste *Classes*, puis un membre.

Une fois qu'un élément apparaît tout en bas, en vous avez déjà une description sommaire et, si vous tapez F1, vous aurez un écran d'aide sur cet élément.

RÉCUPÉRATION DES ERREURS

Il est très mauvais d'avoir un programme qui s'arrête sur une erreur, surtout s'il s'agit d'un développement pour un client car les messages du système sont culpabilisants et laissent entendre qu'il y a une erreur de programmation. VBA permet au programme de garder le contrôle en cas d'erreur.

- Juste avant l'instruction (ou le groupe d'instructions) où l'erreur risque de se produire, implantez `On Error GoTo <étiquette>`. Après le groupe, implantez `On Error GoTo 0`.
- Après l'étiquette, on implante la routine de traitement de l'erreur. Elle contient sûrement des instructions `MsgBox` qui préviennent de l'erreur et sont moins rebutantes que le message normal du système.
- En principe, on sait quelle est l'erreur produite puisqu'on connaît les instructions qui suivent le `On Error`… Toutefois, on peut tester `Err.Number` pour vérifier que c'est l'erreur prévue. Par exemple 11 est le numéro correspondant à la division par 0. `Err.Description` est une chaîne décrivant l'erreur.
- La routine doit se terminer par une instruction `Resume` :
  - `Resume` (tout court) fait revenir à l'instruction qui a causé l'erreur. Il faut donc que le traitement ait résolu le problème, sinon, elle se reproduit.
  - `Resume Next` fait revenir à l'instruction qui suit celle qui a causé l'erreur. Donc le traitement remplace celle-ci, ou on y renonce.
  - `Resume <étiquette>` (rarement employé) fait sauter à l'étiquette indiquée.
  - N'oubliez pas d'implanter un `Exit Sub` juste avant l'étiquette du traitement d'erreur, sinon, on tombe inopinément sur ce traitement.

Exemple : On essaie d'ouvrir un classeur ; en cas d'impossibilité, on demande à l'utilisateur de fournir la bonne désignation du fichier. Le retour se fait sur l'instruction d'ouverture, puisque l'erreur est censée être corrigée.

```
Sub Ouvrir()
Dim FN As String
  FN = "C:\ClasseurA.xlsx"
  On Error GoTo TraitErr
  Workbooks.Open Filename:=FN
  On Error GoTo 0
…
…
  Exit Sub
TraitErr:
  FN = InputBox("Impossible d'ouvrir " + FN + _
    vbCr + "Entrez la bonne désignation")
  Resume
End Sub
```

Il y a une autre version page 90 et un autre exemple à la fin du chapitre 5 (page 86).